

A DDE Macro to Put Data Anywhere in Excel

Ting Sa, Cincinnati Children's Hospital Medical Center, Cincinnati, OH

Shiran Chen, Cincinnati Children's Hospital Medical Center, Cincinnati, OH

ABSTRACT

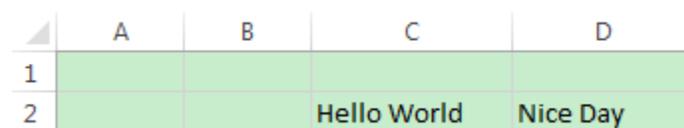
In this paper, the authors introduce a SAS macro that integrates the SAS DDE techniques so you can use this macro to put data anywhere in your Excel file. SAS users only need to prepare the input data and call the macro for the data to be automatically inserted into the Excel file. This macro will be very helpful if you have large amount of data to be inserted into different cells in the Excel file.

INTRODUCTION

Dynamic Data Exchange (DDE) is a method available in SAS that you can use to insert data into any locations in the Excel file. The example below shows how to use DDE to insert the strings "Hello World" and "Nice Day" into the Excel sheet "test sheet" in the Excel file "c:\test.xlsx". The "Hello World" string will be inserted into cell (2, 3) and the string "Nice Day" will be inserted into cell (2, 4). After it inserts the two strings, the program saves the results as the new Excel file "c:\test1.xls".

```
options noxwait noxsync;
*open the excel file, you must create the "c:\text.xlsx" file first before
you run the program;
%sysexec "C:\test.xlsx";
*create filename f and let it point to the area contains cell(2,3) and
cell(2,4) in the excel sheet 'test sheet' of the excel file 'C:\test.xlsx';
*the notab option is used so that a string containing blanks will be added
into one cell, without this option, each word of a string is stored in a
single cell;
filename f DDE "excel|test sheet!r2c3:r2c4" notab;
*write "Hello World" to the cell(2,3) and "Nice Day" to the cell(2,4) in the
excel sheet;
data _null_;
  file f;
  a = "Hello World";
  b = "Nice Day";
  *put 20 here means each cell in the excel can hold up to 20 characters;
  *'09'x is the hexadecimal value for tab, using it to separate each variable;
  put a $20. '09'x b $20.;
run;
*save the final output excel file as "c:\test1.xls";
filename commands DDE 'EXCEL|SYSTEM';
data _null_;
  file commands;
  put "[Save.as("""c:\test1.xls""")]";
  put "[Close(0)]";
run;
```

Figure 1 is the screenshot of the "c:\test1.xls" file.



	A	B	C	D
1				
2			Hello World	Nice Day

Figure 1. Screenshot for the new "c:\test1.xls" file

If you have a lot of data to be inserted into different places in the Excel file, it will cost a lot of time to write the DDE codes. Using the macro described in this paper, the user just needs to prepare the input SAS data set and call the macro to generate the desired Excel file. Figure 2 displays a sample SAS input data set that the macro can use. The “text” column contains the data you want to insert into the Excel file. The “sheetnm” column contains the Excel sheet name information. The “rowno” contains the row number of the cell and the “colno” contains the column number of the cell. For example, the first row in the Figure 2 sample data set tells the macro to insert string “Average Squared Error” to the cell (1,1) in the Excel sheet “Sheet1”.

	text	sheetnm	rowno	colno
1	Average Squared Error	Sheet1	1	1
2	Unable to determine the code variable dependencies.	Sheet2	2	2
3	Overall Precision Rate	Sheet3	3	3
4	CR Cutoff	Sheet1	4	4
5	CR Sample Depth	Sheet2	5	0
6	Intermediate file creation failed.	Sheet3	6	1

Figure 2. Screenshot of an input SAS data set for the macro

THE ADDVALUETOEXCEL MACRO

Below are the SAS macro codes:

```
%macro addValueToExcel(libnm=, datanm=, in_excelnm=, out_excelnm=);
*open the excel file;
options noxwait noxsync;
%sysexec "&in_excelnm.";

*create the tmp data set to copy the data from the input SAS data set;
data tmp;
set &libnm..&datanm.;
run;

*do the sorting so that it could improve the program efficiency;
proc sort data=tmp;by sheetnm rowno colno;run;

*create the "codes" variable so that it will save the SAS DDE codes;
data tmp;
set tmp;
length codes $500.;
if not missing(text) then do;
codes=cats(" ", "filename", "cellval", "DDE",
cats("'excel|", sheetnm, "!r", rowno, "c", colno, ":r", rowno, "c", colno, "'"), "notab;
",
"data _null_;file cellval;", cats("cellval=", text, ";"),
"put cellval $50. '09'x;", "run;");
end;
run;

*execute the SAS DDE codes saved in the "codes" variable;
data _null_;
set tmp;
call execute(codes);
run;

*delete the "tmp" data set;
```

```

proc datasets;delete tmp;run;quit;

*saves the final excel file;
filename commands DDE 'EXCEL|SYSTEM';
data _null_;
file commands;
put "[Save.as("""&out_excelnm.""")]";
put "[Close(0)]";
run;
%mend;

```

- The “libnm” macro variable saves the library name of the input SAS dataset name.
- The “datanm” macro variable saves the SAS input data set name.
- The “in_excelnm” macro variable saves the Excel file name you want to insert data into. This file must be created first before you call the macro; you also need to ensure that this file contains the sheets that are listed in the input SAS data set.
- The “out_excelnm” macro variable saves the name of the final output Excel file.

CALL THE MACRO

The following codes will create a sample data set “test_data” that can be used as the input data set by the macro.

```

data test_data;
  set sashelp.aacomp(keep=text firstobs=1 obs=100);
  if mod(_n_,3)=1 then sheetnm="Sheet1";
  else if mod(_n_,3)=2 then sheetnm="Sheet2";
  else if mod(_n_,3)=0 then sheetnm="Sheet3";
  rowno=_n_;
  colno=mod(_n_,5);
run;

```

Using the following SAS codes to call the macro, the macro will insert all the data from the “test_data” data set to the “C:\test1.xlsx” Excel file and then save the final output Excel file as the “C:\test_new.xlsx”. Figure 3 shows part of the Excel sheet “Sheet1” in the “C:\test_new.xlsx” file.

```

%addValueToExcel(libnm=work,datanm=test_data,in_excelnm=%str(C:\test1.xlsx),
out_excelnm=%str(C:\test_new.xlsx));

```

	A	B	C	D	E	F	G	H	I
1	Average Squared Error								
2									
3									
4				CR Cutoff					
5									
6									
7				Maximum Classification Rate (CR)					
8									
9									
10									
11									
12									
13				Cumulative Percentage of Nonevents					
14									
15									
16				Cumulative Percentage of Events					

Figure 3. Part of the “Sheet1” in the “C:\test_new.xlsx” file.

HOW THE MACRO WORKS

Here is a step-by-step explanation of the macro:

1. The following SAS codes will open the Excel file you will insert the data into. You must ensure that this Excel file has already been created before you call the macro, and the file should contain the sheets that are included in the input SAS data set.

```
options noxwait noxsync;
%sysexec "&in_excelnm.";
```

2. The following SAS codes are used to first copy all the data from the input SAS data set to the “tmp” SAS data set. Then the macro sorts the “tmp” SAS data set by sheet name, row number and column number; by doing this, the program can run more efficiently.

```
data tmp;
set &libnm.&datanm.;
run;

proc sort data=tmp;by sheetnm rowno colno;run;
```

3. The following SAS codes create the “codes” variable in the “tmp” data set. The “codes” column is used to save the SAS DDE codes. Figure 4 displays an example of the DDE codes that have been saved in the “codes” column.

```
data tmp;
set tmp;
length codes $500.;
if not missing(text) then do;
codes=catx(" ", "filename", "cellval", "DDE",
cats("'excel|", sheetnm, "!r", rowno, "c", colno, ":r", rowno, "c", colno, "'"), "notab;
",
"data _null_;file cellval;", cats("cellval=", text, ";"),
"put cellval $50. '09'x;", "run;"); *You can change 50 to other number so that
the excel cell can hold the number of characters you want;
end;
run;
```

codes
filename cellval DDE 'excel Sheet1!r1c1:r1c1' notab; data _null_;file cellval; cellval='Average Squared Error'; put cellval \$50. '09'x; run;

Figure 4. An example of the DDE codes that have been created and saved in the “codes” column

Currently, in the macro, each Excel cell can contain up to 50 characters, but this number can be changed in the macro.

4. The following codes execute the DDE codes saved in the “codes” column and delete the tmp data set;

```
data _null_;
set tmp;
call execute(codes);
run;
proc datasets;delete tmp;run;quit;
```

5. The following SAS codes save the file as the final output Excel file.

```
filename commands DDE 'EXCEL|SYSTEM';
data _null_;
file commands;
put "[Save.as("&out_excelnm."")]";
put "[Close(0)]";
run;
```

CONCLUSION

The macro presented in this paper is a helpful tool to insert data to Excel files, especially if the data size is large and the data have to be placed in different cells. You can further extend the functions of the macro by also inserting data to multiple Excel files. DDE can additionally insert graphs and images into the Excel files, so you can upgrade the macro codes to also implement those functionalities.

ACKNOWLEDGMENTS

The authors wishes to thank the Division of Biostatistics and Epidemiology at Cincinnati Children's Hospital Medical Center for its support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Ting Sa
Division of Biostatistics and Epidemiology, Cincinnati Children's Hospital Medical Center
5136363674
Ting.Sa@CCHMC.ORG

Shiran Chen
Division of Biostatistics and Epidemiology, Cincinnati Children's Hospital Medical Center
5136367124
Shiran.Chen@CCHMC.ORG

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.